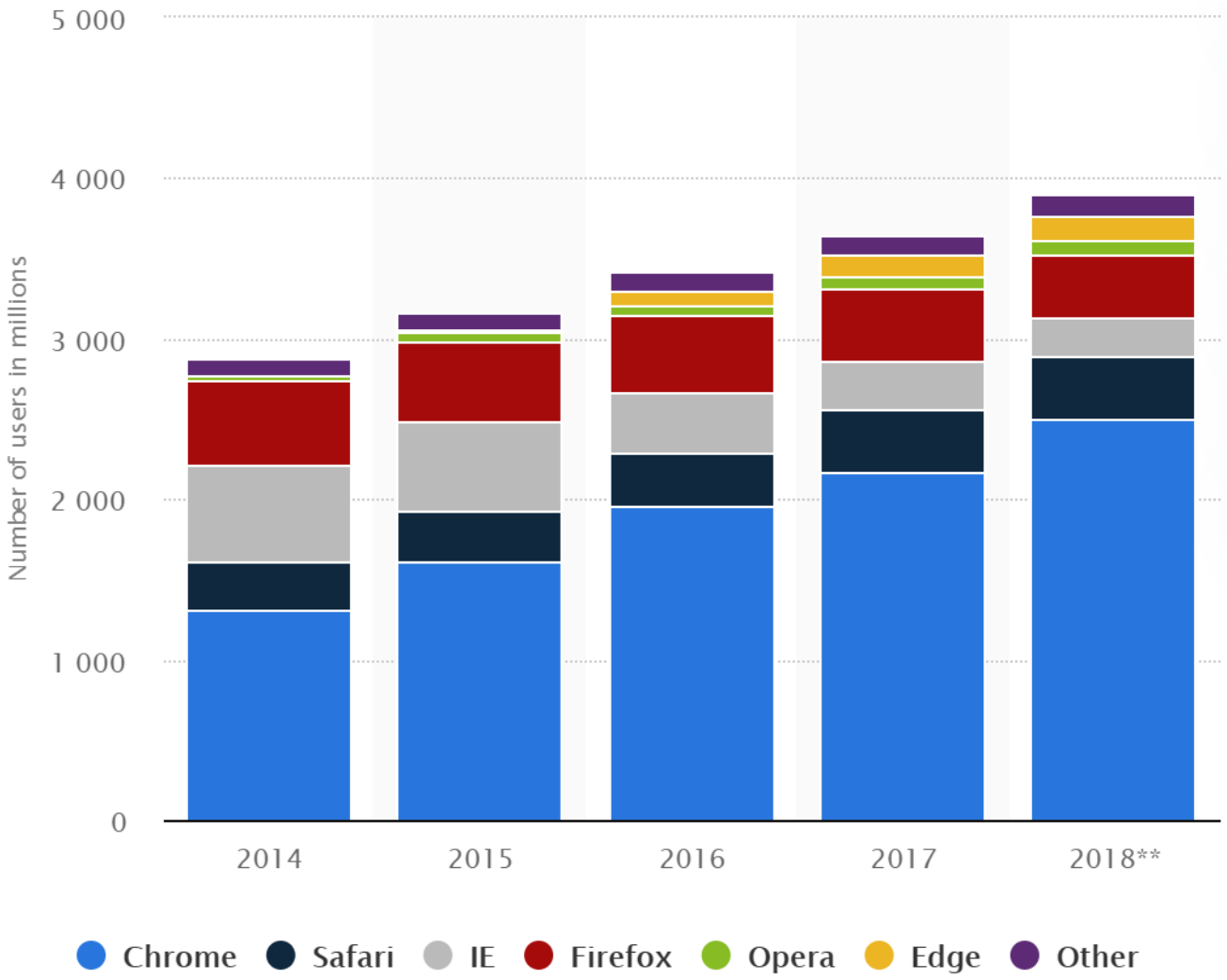
Montage: A Neural Network Language Model-Guided JavaScript Engine Fuzzer

Suyoung Lee, HyungSeok Han, Sang Kil Cha, Sooel Son

KAIST

Popularity of Web Browsers



4 billion users

Source: https://www.statista.com/statistics/543218/worldwide-internet-users-by-browser/

JS Engine Vulnerabilities

Costs up to **$500,000**

Critical security threat!

Source: https://zerodium.com/program.html

JS Engine Fuzzing

```
let v0 = new Array(0x10000);

v0 = v0.fill(0x1234).join(', ');

eval('new Array(' + v0 + ')');
```

How can a fuzzer generate JS inputs?

Previous Work

1. Mutation-based fuzzers

- LangFuzz, IFuzzer, and GramFuzz
- Combining **AST subtrees** extracted from JS seeds

2. Generation-based fuzzers

- jsfunfuzz
- Applying **JS grammar rules** from scratch

Previous Work – Building Blocks

1. Mutation-based fuzzers

- LangFuzz, IFuzzer, and GramFuzz
- Combining **AST subtrees** extracted from JS seeds

2. Generation-based fuzzers

- jsfunfuzz
- Applying **JS grammar rules** from scratch

Previous Work – Building Blocks

Current AST

All appendable building blocks

They **randomly select** building blocks!

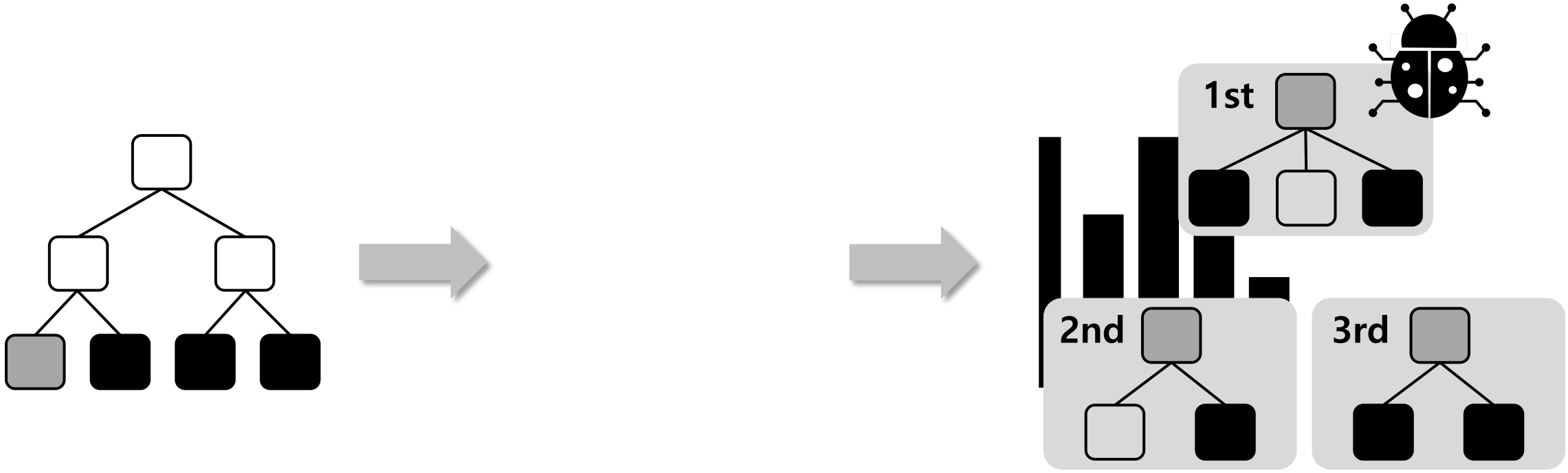
Patterns of Bug-triggering Code

```
let v0 = new Array(0x10000);

v0 = v0.fill(0x1234).join(', ');

eval('new Array(' + v0 + ')');
```

Bug-triggering JS code

Extracted Patterns

Patterns of Bug-triggering Code



Current AST

Extracted Building blocks

Which building block is **more likely to** trigger JS engine bugs?

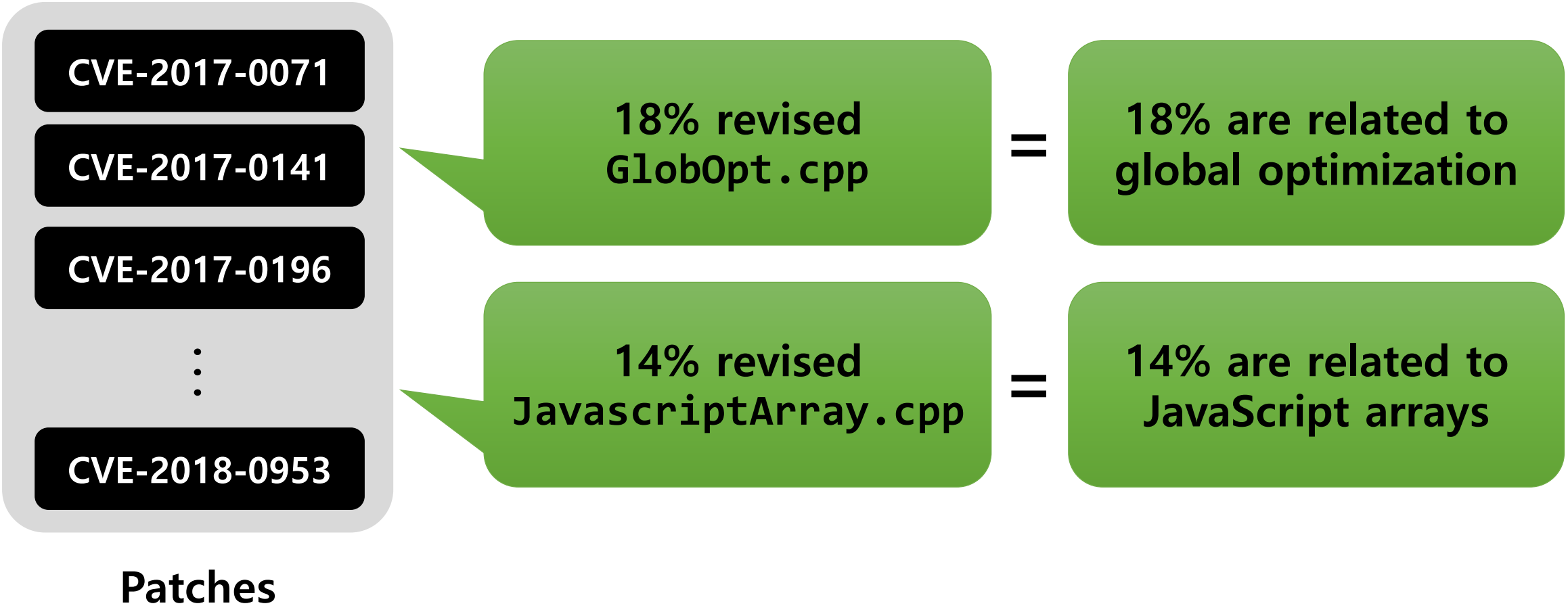
Study on JS Engine Vulnerabilities

1. **Functional commonalities**

2. **Syntactical commonalities**

Functional Commonalities

- **Study 1. Patches of 50 ChakraCore CVEs**



Syntactical Commonalities

- Study 2. AST subtrees from two sets

At August, 2016

Extracted subtrees

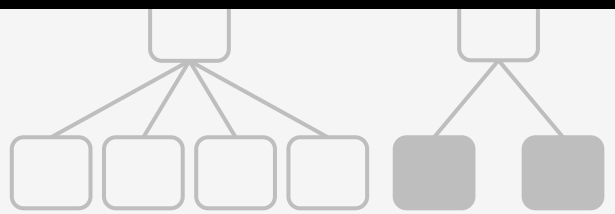
After August, 2016

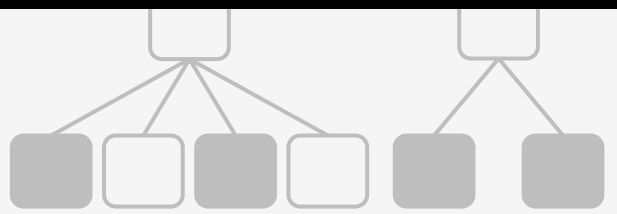
Extracted subtrees

95% already existed!

JS Test 2038

CVE-2018-0980





2,038 JS tests from ChakraCore repo

67 PoCs triggering ChakraCore CVEs

Syntactical Commonalities

- **Study 2. AST subtrees from two sets**

```
function f0() {
    'use asm';
    const v0 = Math.fround(1);
    function f1() {
        var v1 = v0;
        var v2 = Math.fround(4);
    }
    return {f1: f1}
}
```

JS test from ChakraCore repo

Syntactical Commonalities

- **Study 2. AST subtrees from two sets**

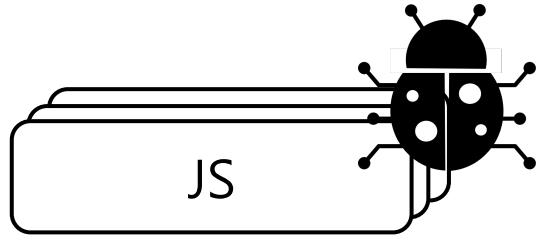
```
function f0() {
    'use asm';
    const v0 = 1.0;
    function f1() {
        var v1 = v0;
        var v0 = 0;
    }
    return f1;
}
```

PoC (CVE-2017-11911)

Our Approach

1. Functional commonalities

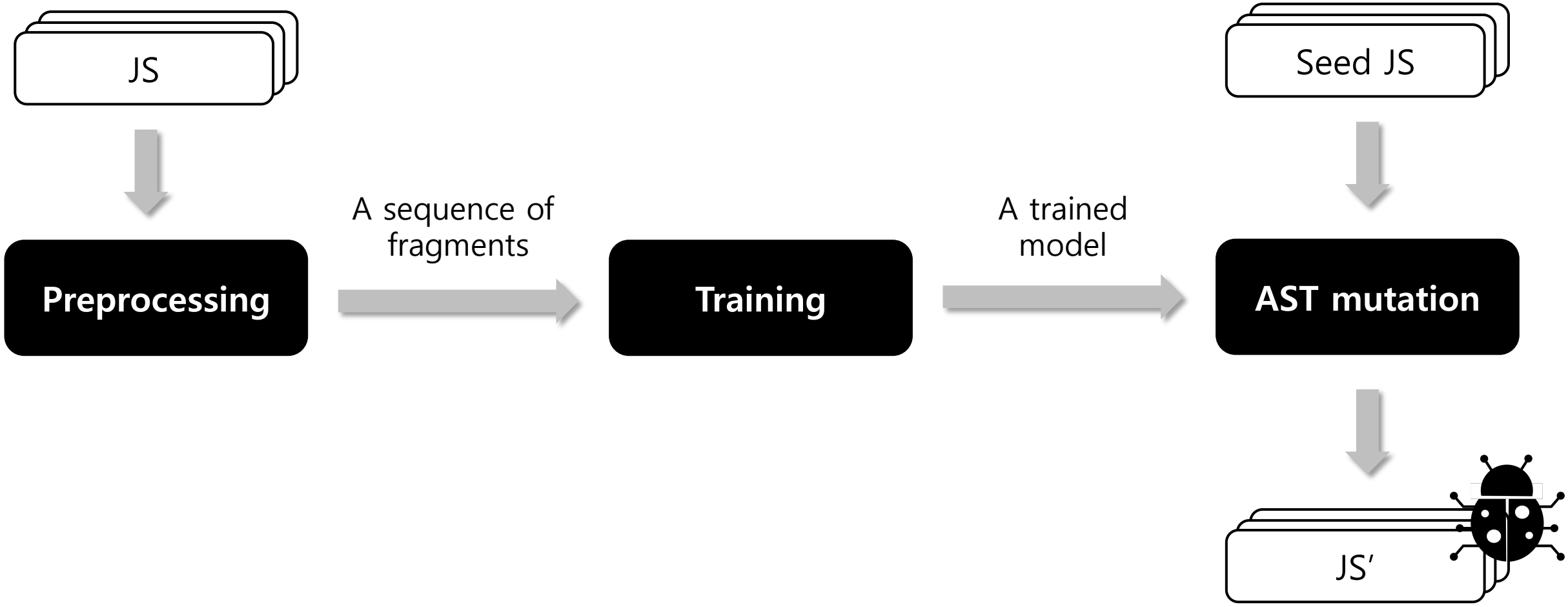
- Mutating existing JS regression tests



2. Syntactical commonalities

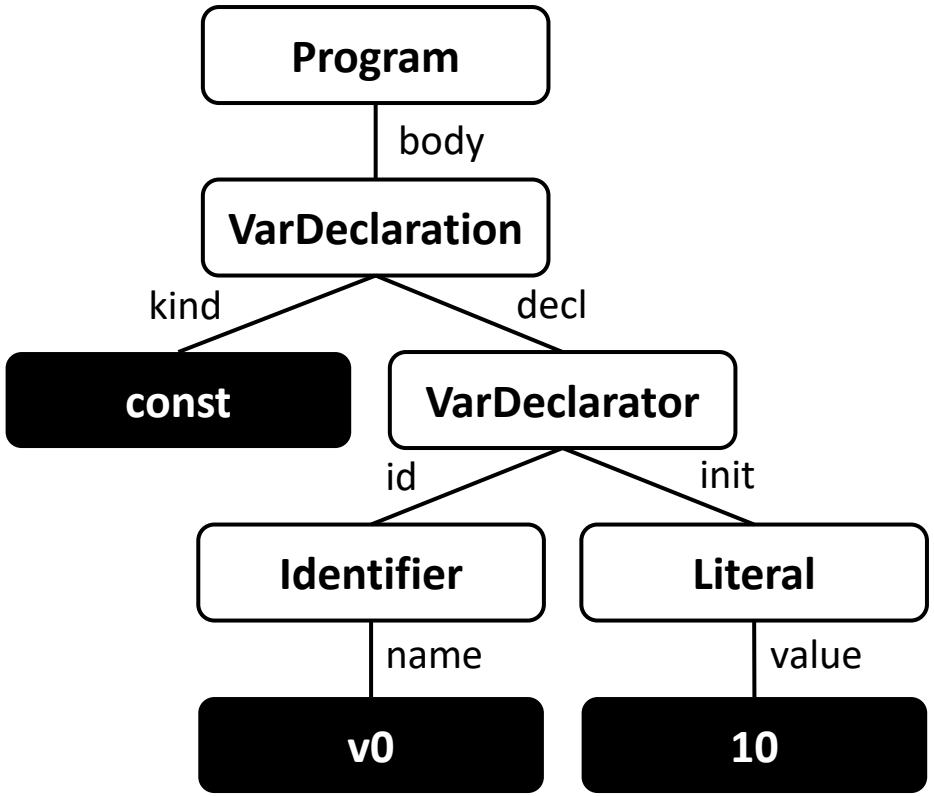
- Modeling of the relationships between AST subtrees

Montage Overview



Preprocessing

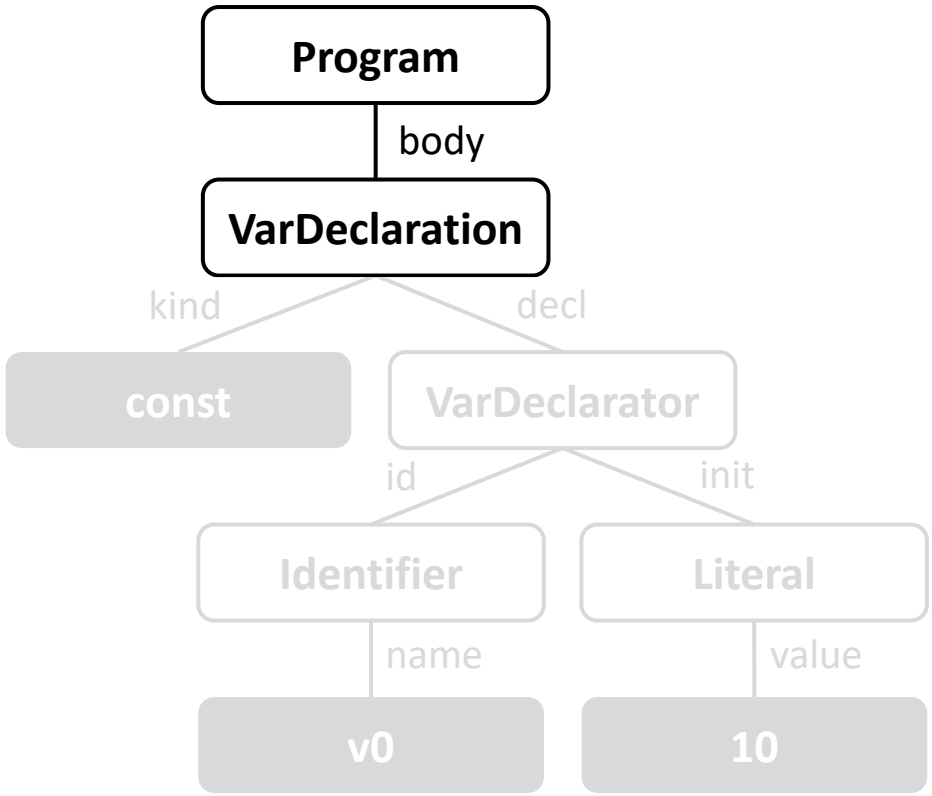
```
const v0 = 10;
```



JavaScript AST

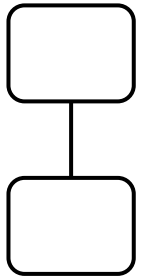
Preprocessing

```
const v0 = 10;
```



JavaScript AST

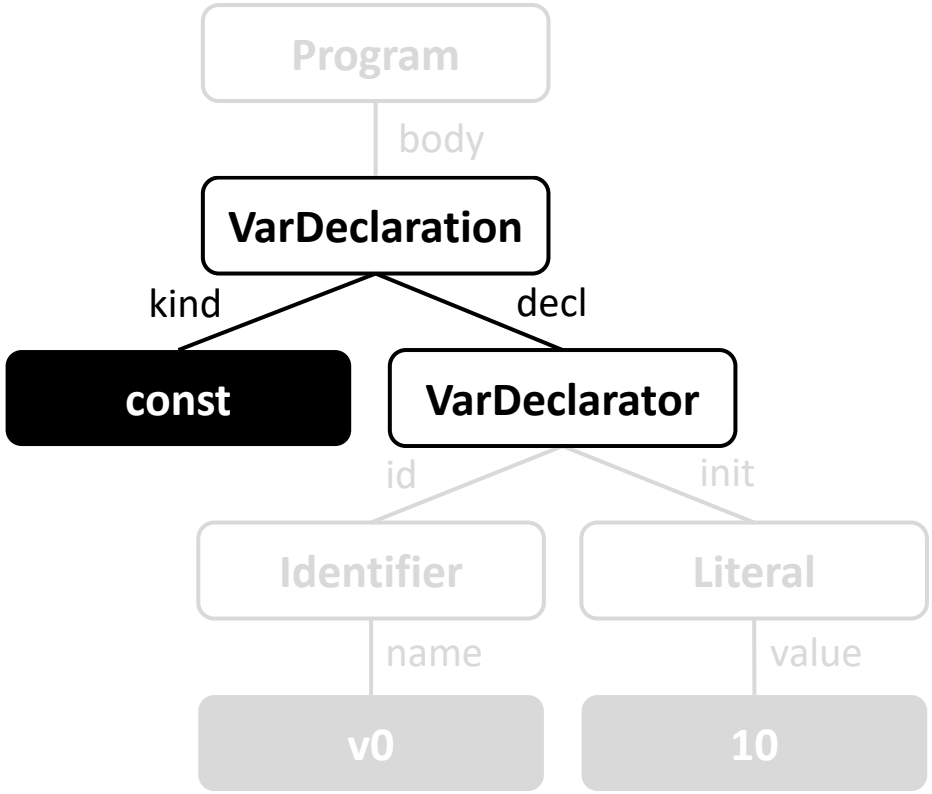
Pre-order traversal



A sequence of fragments

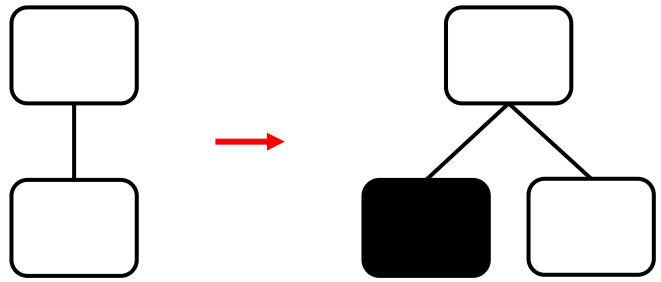
Preprocessing

```
const v0 = 10;
```



JavaScript AST

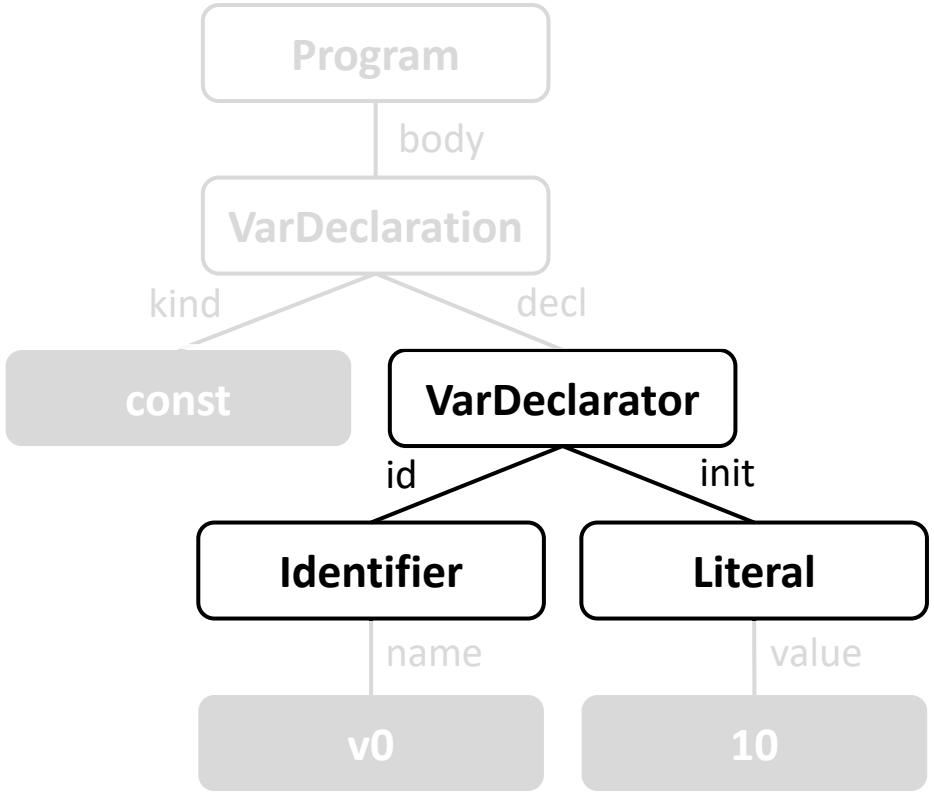
Pre-order traversal



A sequence of fragments

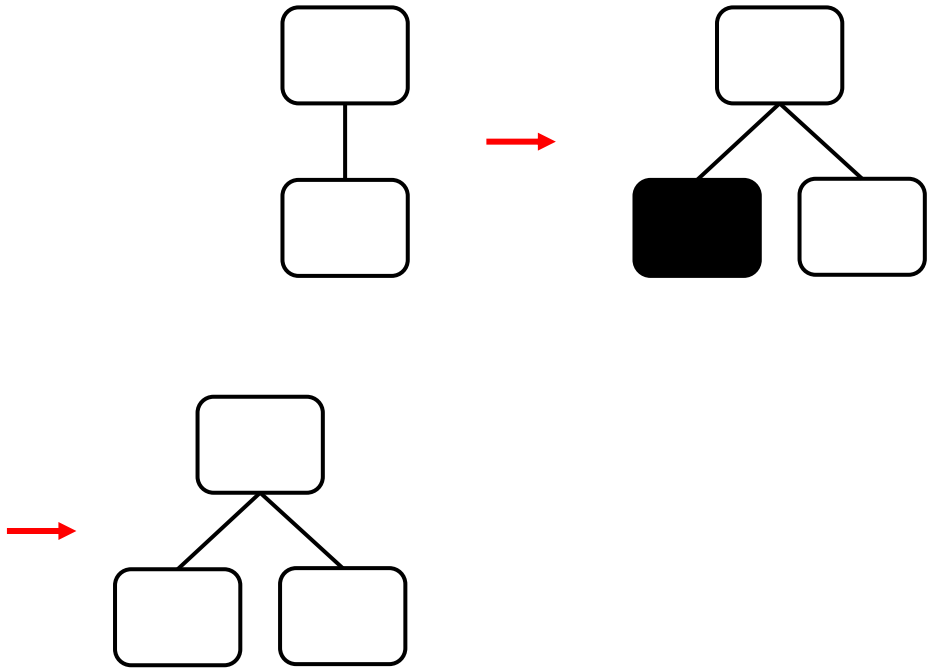
Preprocessing

```
const v0 = 10;
```



JavaScript AST

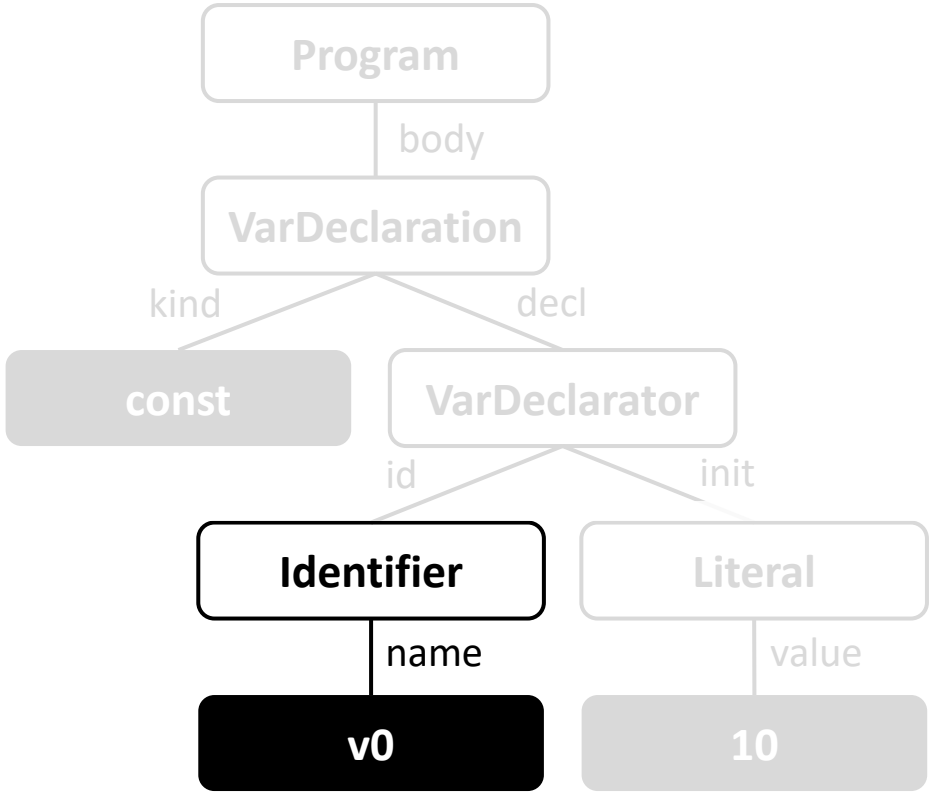
Pre-order traversal



A sequence of fragments

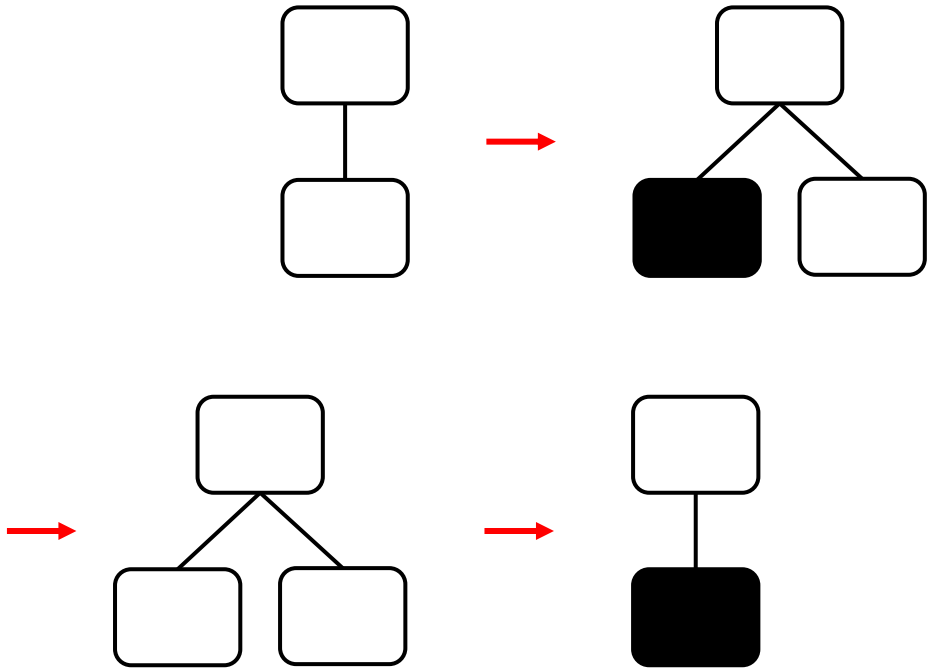
Preprocessing

```
const v0 = 10;
```



JavaScript AST

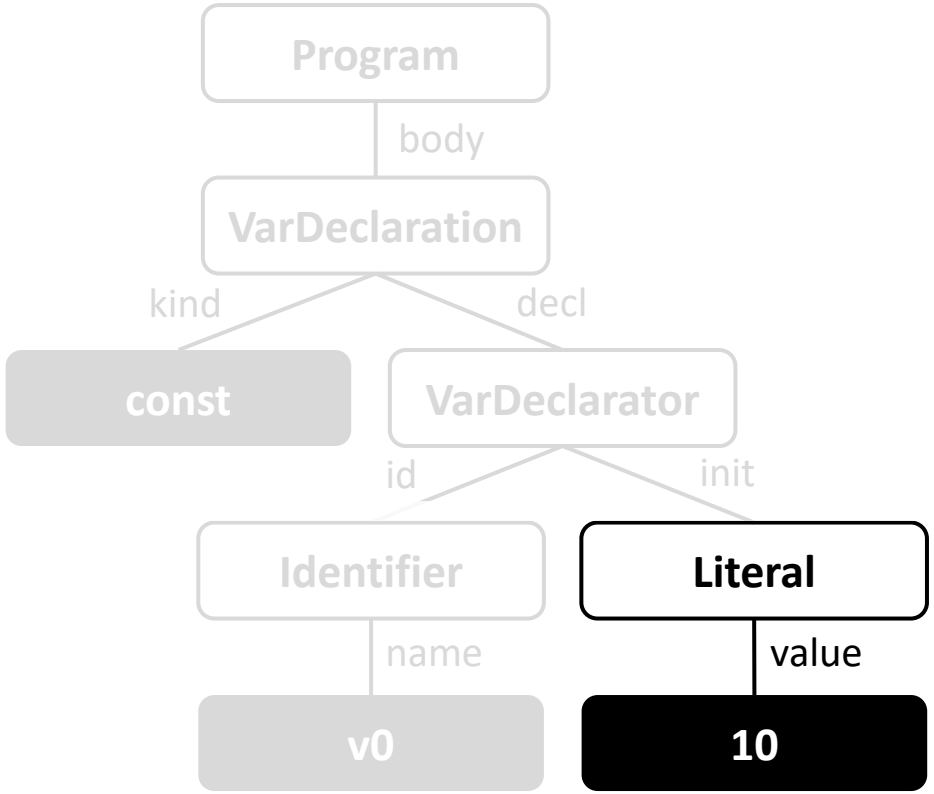
Pre-order traversal



A sequence of fragments

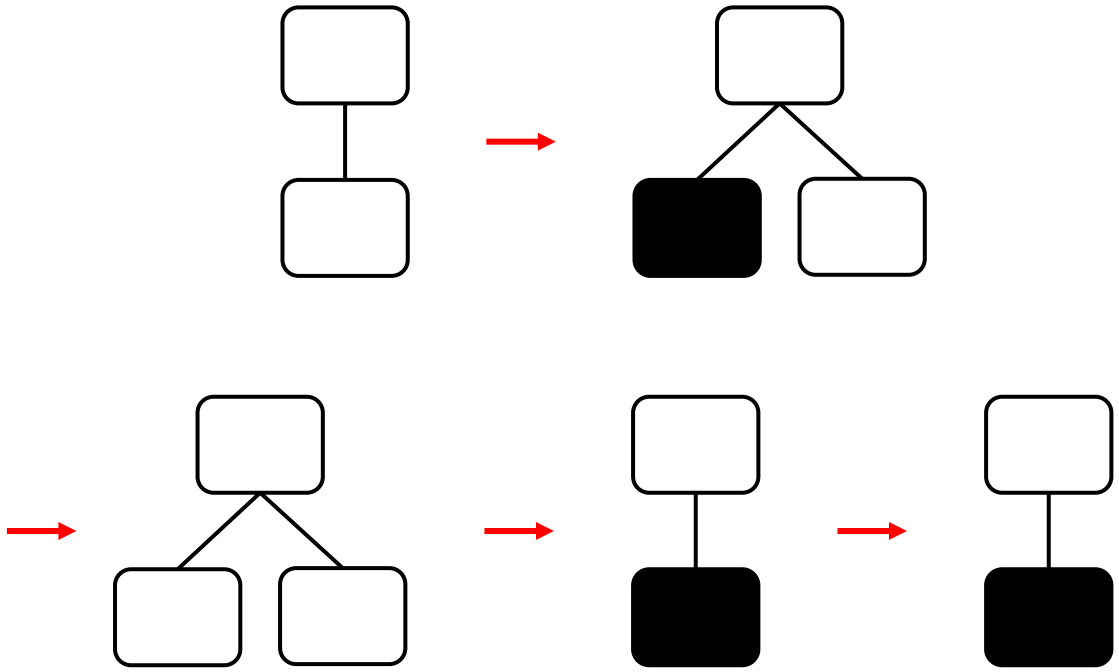
Preprocessing

```
const v0 = 10;
```



JavaScript AST

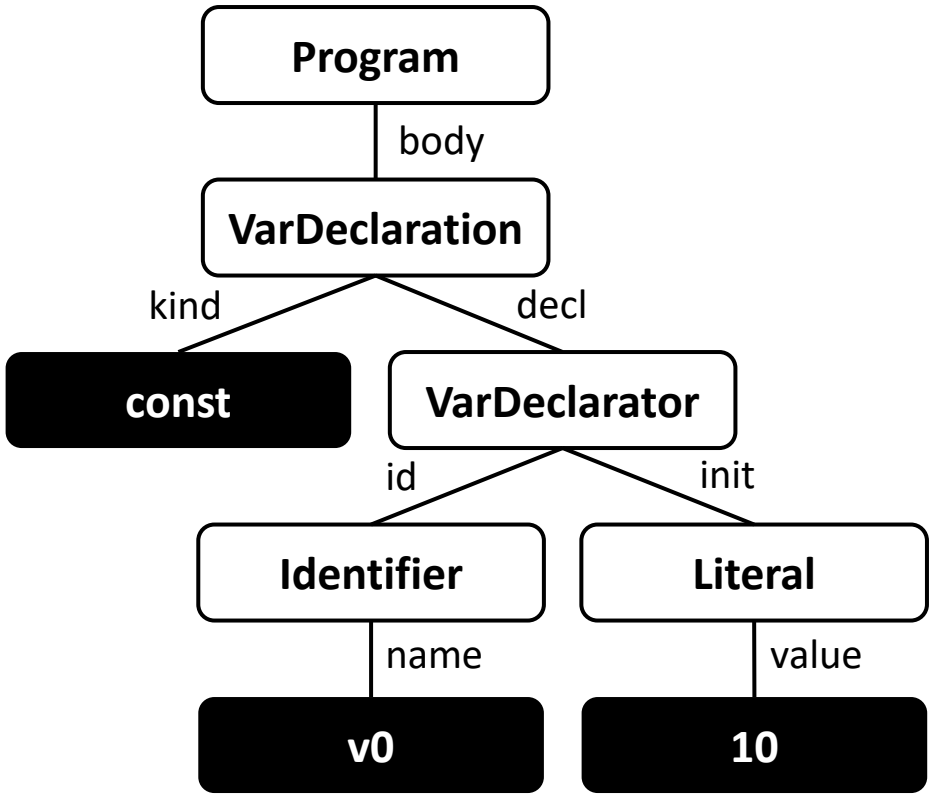
Pre-order traversal



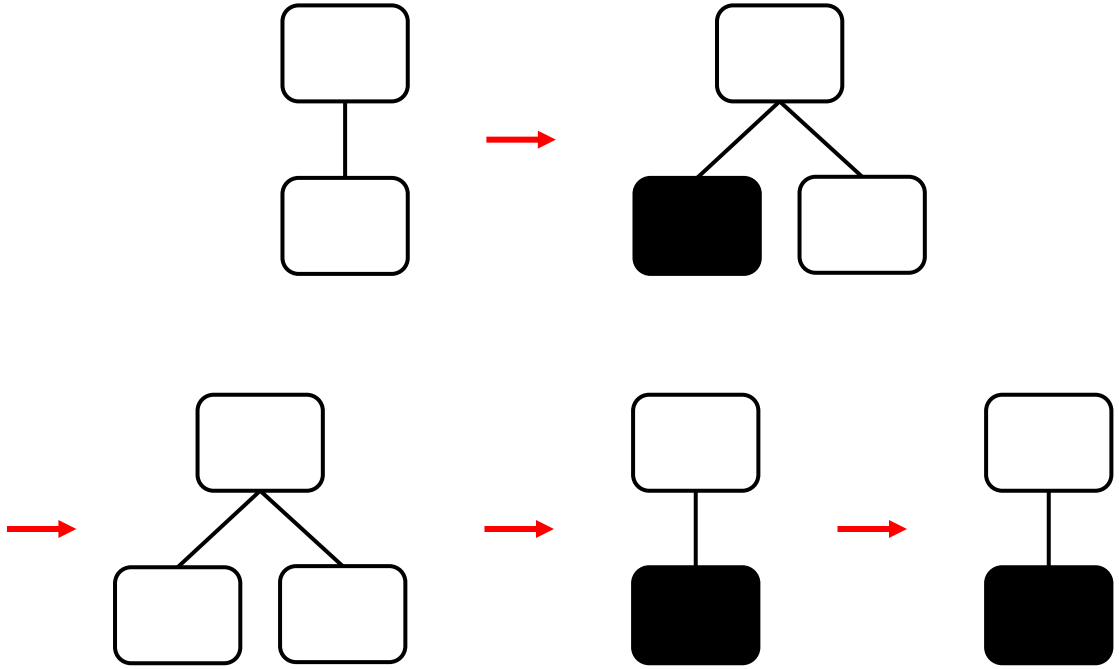
A sequence of fragments

A Sequence of Fragments

```
const v0 = 10;
```



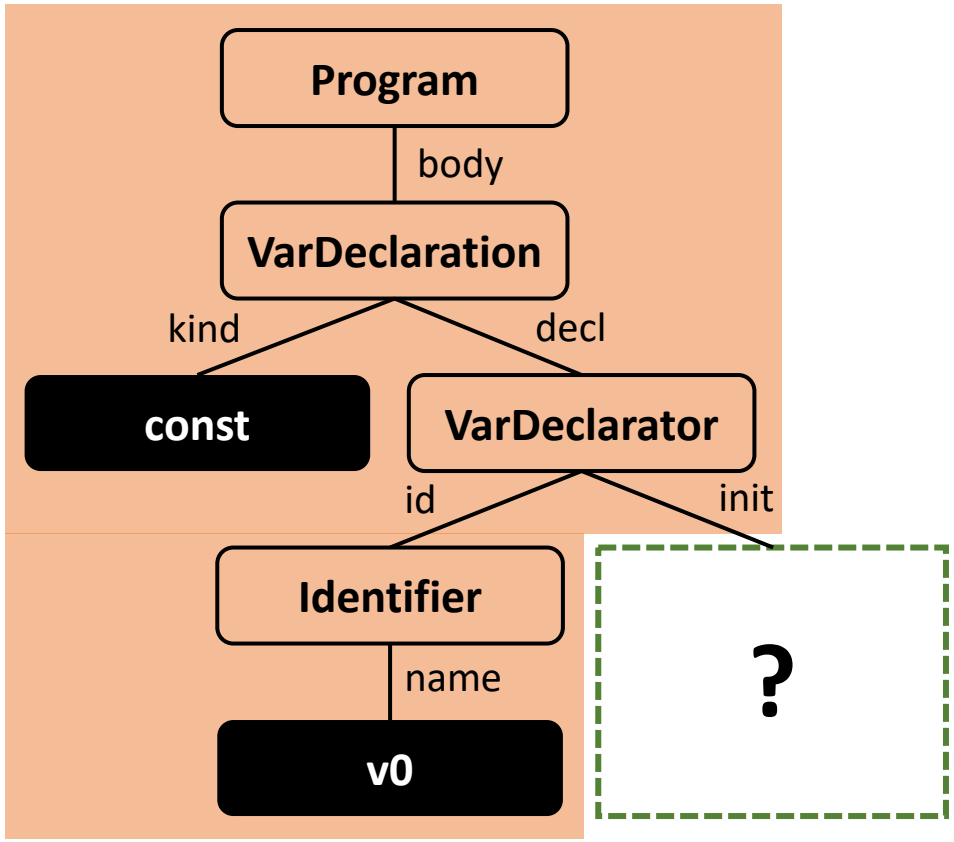
JavaScript AST



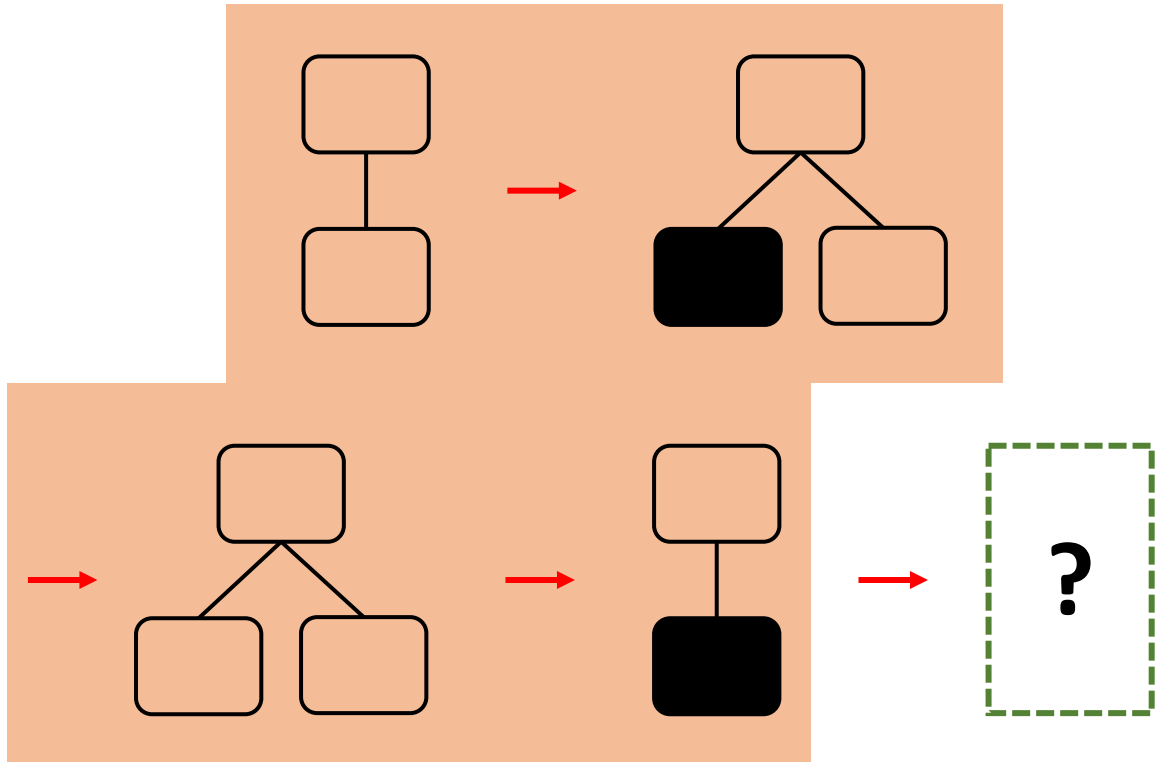
A sequence of fragments

Global Compositional Relationship

```
const v0 = 10;
```



JavaScript AST

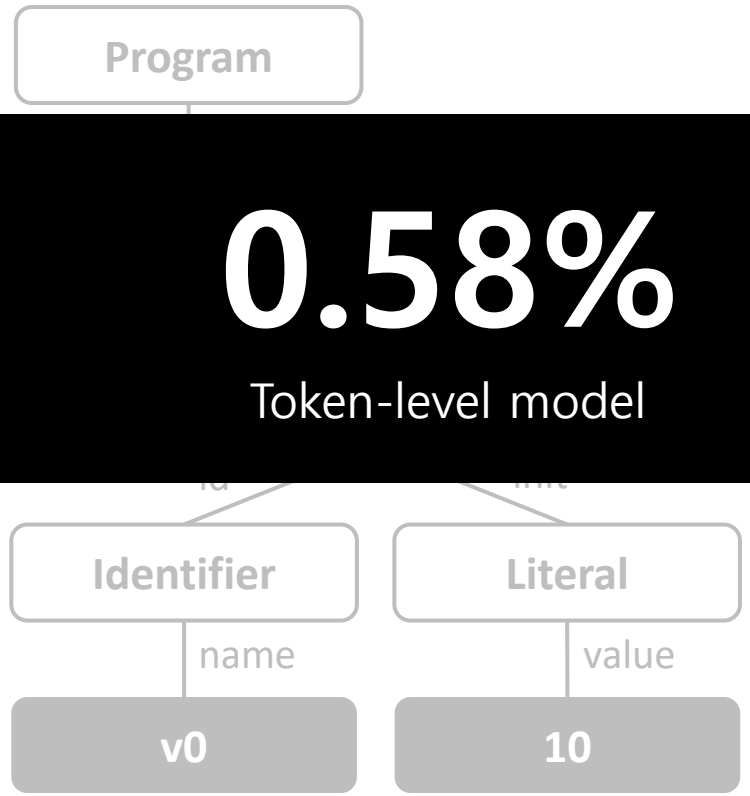


A sequence of fragments

vs. A Sequence of Tokens [1-3]

```
const v0 = 10;
```

% of Valid JS code



JavaScript AST

0.58% Token-level model

vs.

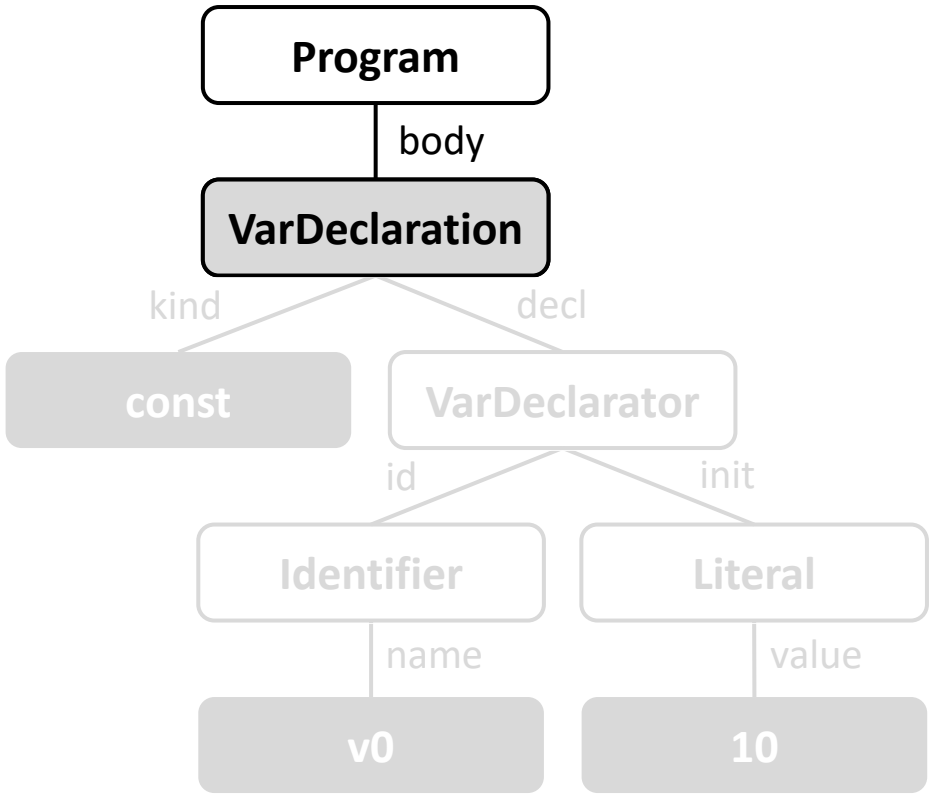
58.26% Fragment-level model

A sequence of tokens

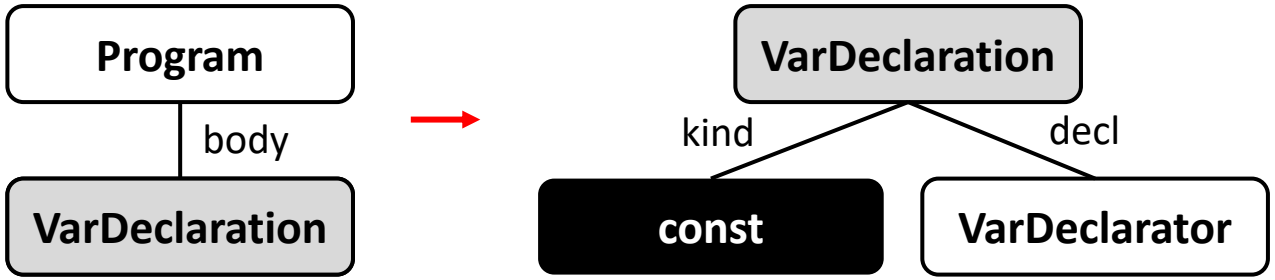
[1] Cummins et al. Compiler fuzzing through deep learning (ISSTA'18).
[2] Godefroid et al. Learn&Fuzz: Machine learning for input fuzzing (ASE'17).
[3] Liu et al. DeepFuzz: Automatic generation of syntax valid C programs for fuzz testing (AAAI'19).

Selecting Valid Fragments

```
const v0 = 10;
```



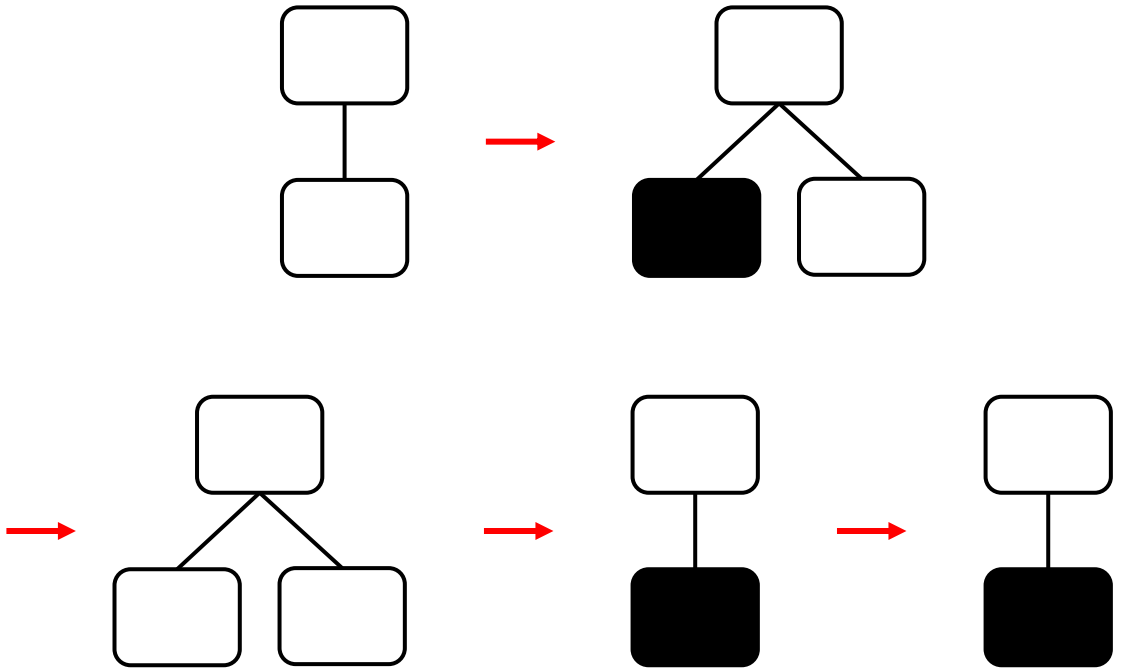
JavaScript AST



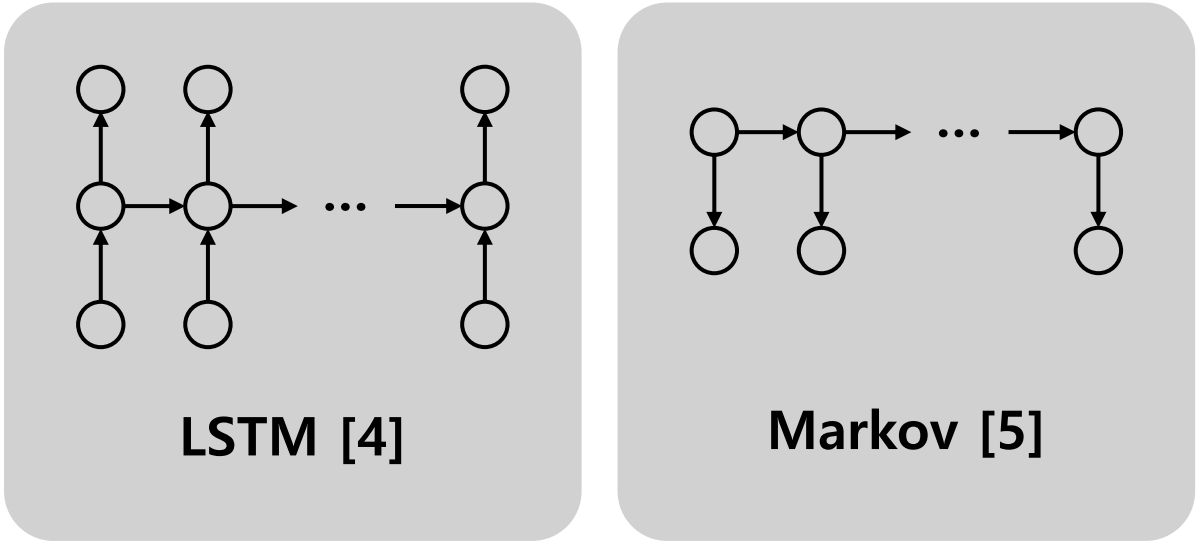
A sequence of fragments

Applicable to Any Language Models

```
const v0 = 10;
```



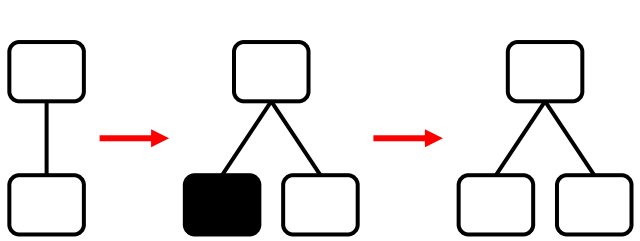
A sequence of fragments



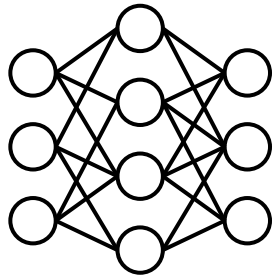
Language models

[4] Hochreiter et al. Long short-term memory. Neural Computation, 9(8), 1997.
[5] Andrey A. Markov. An Example of Statistical Investigation of the Text *Eugene Onegin* Concerning the Connection of Samples in Chains, 1913.

Training an LSTM model



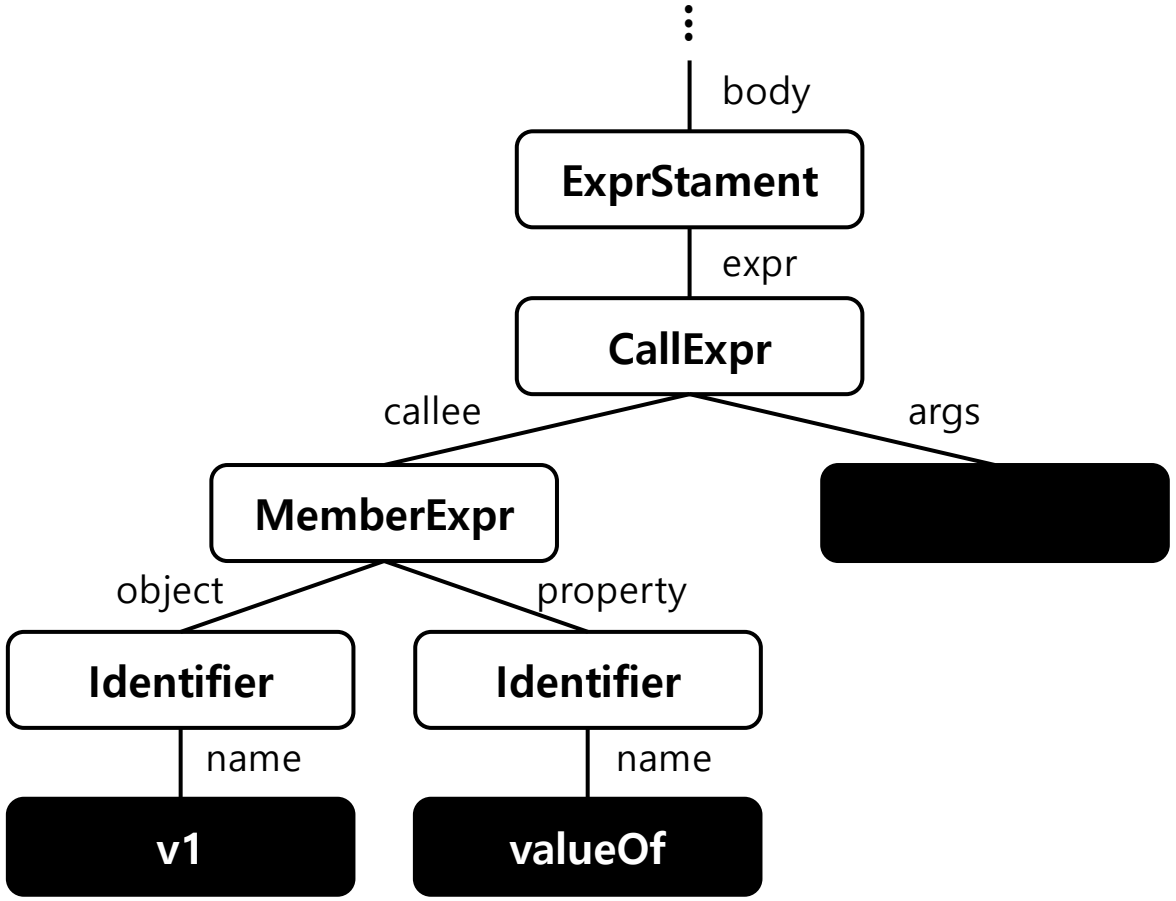
A sequence of preceding fragments



LSTM model

Probability distribution of a next fragment

AST Mutation

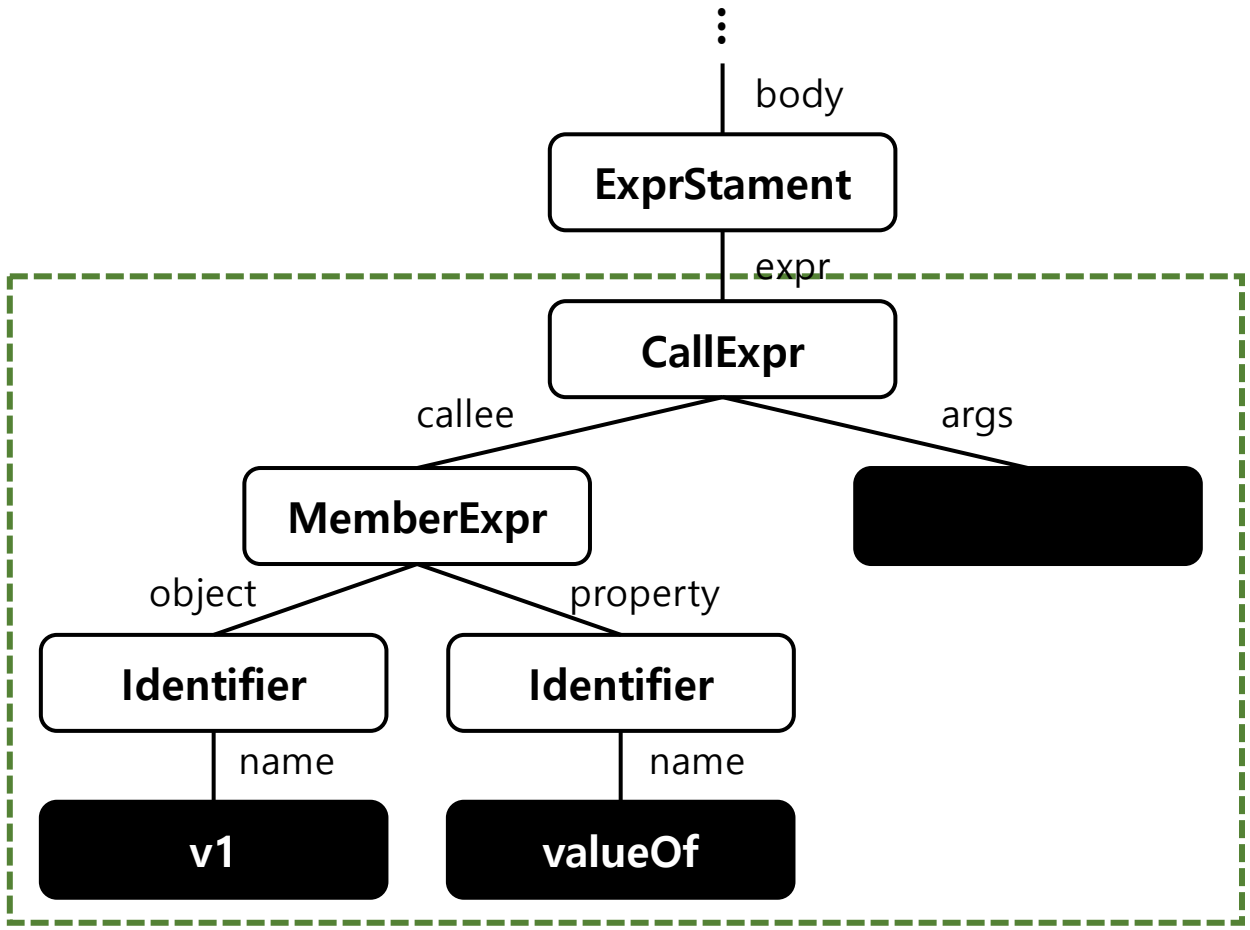


Seed AST

```
var v0 = ‘Hello World’;
v1 = [];
f0();
function f0 () {
    v1.valueOf();
    var v2 = 10;
}
```

Seed JS

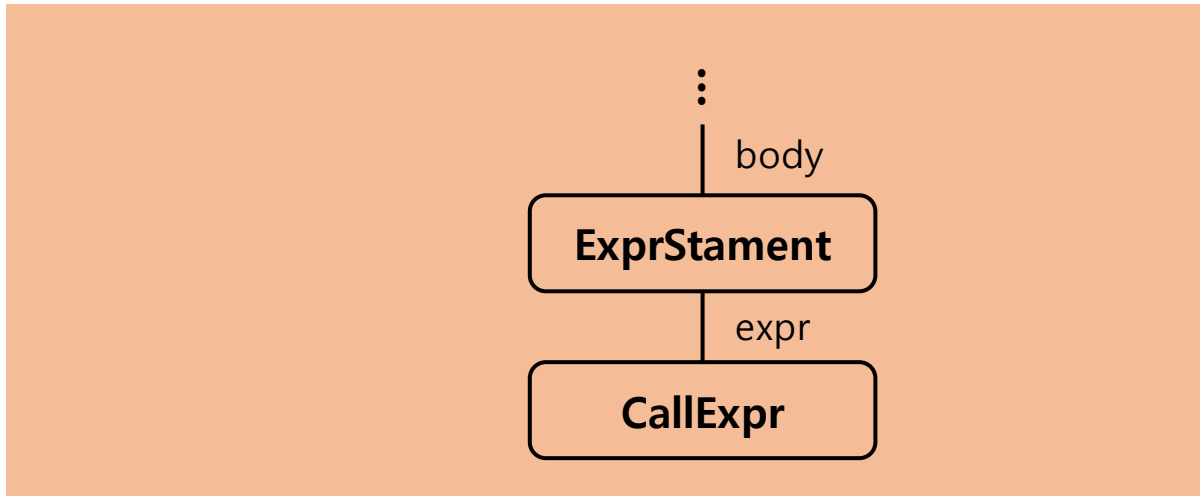
AST Mutation

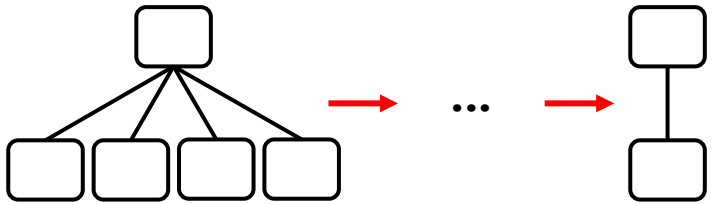


Remove a subtree

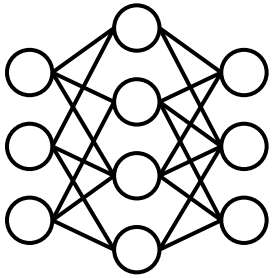
```
var v0 = 'Hello World';
v1 = [];
f0();
function f0 () {
    v1.valueOf();
    var v2 = 10;
}
```

AST Mutation



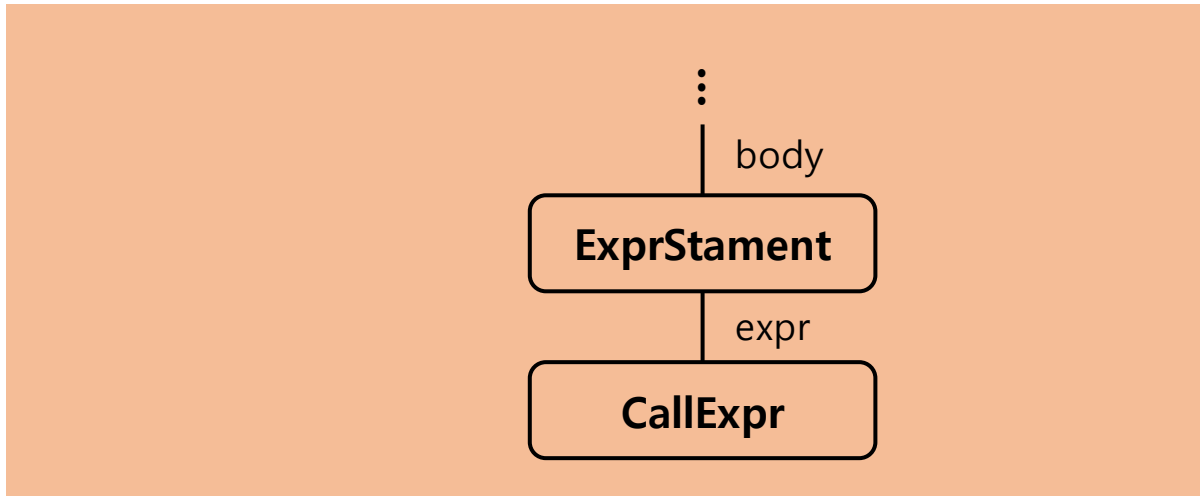


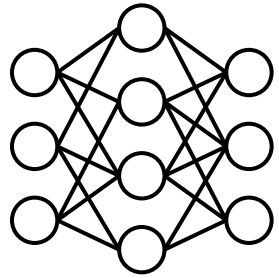
A sequence of fragments representing the current AST



Trained LSTM model

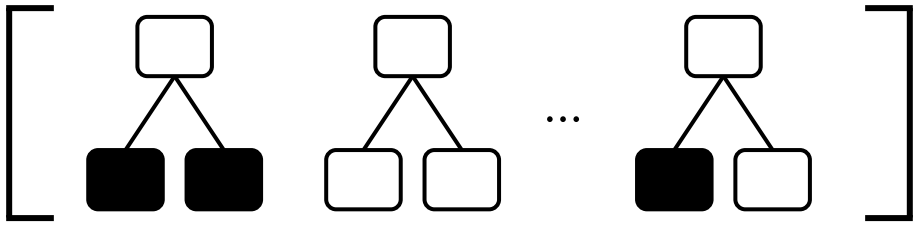
AST Mutation





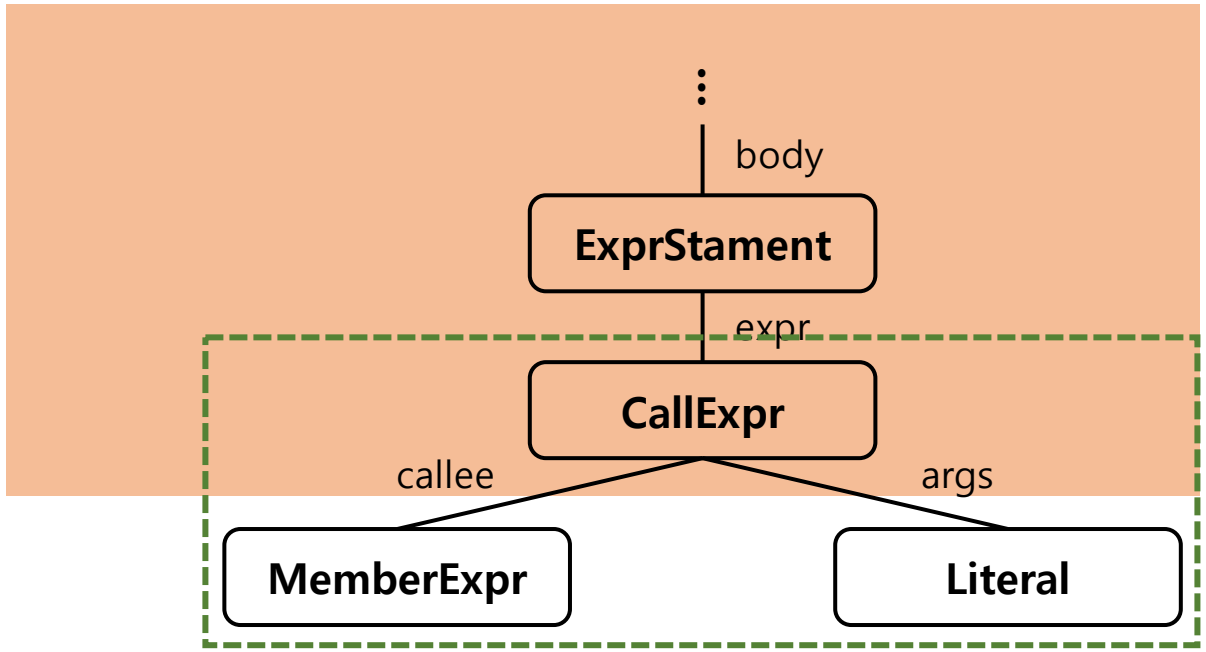
Trained LSTM model

The probability distribution of the next fragment

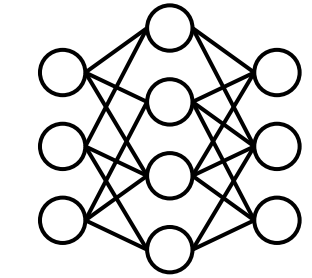


0.76 0.12 0.001

AST Mutation

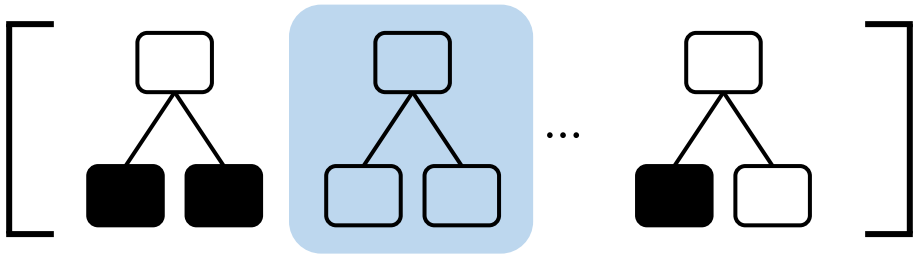


Pre-order traversal



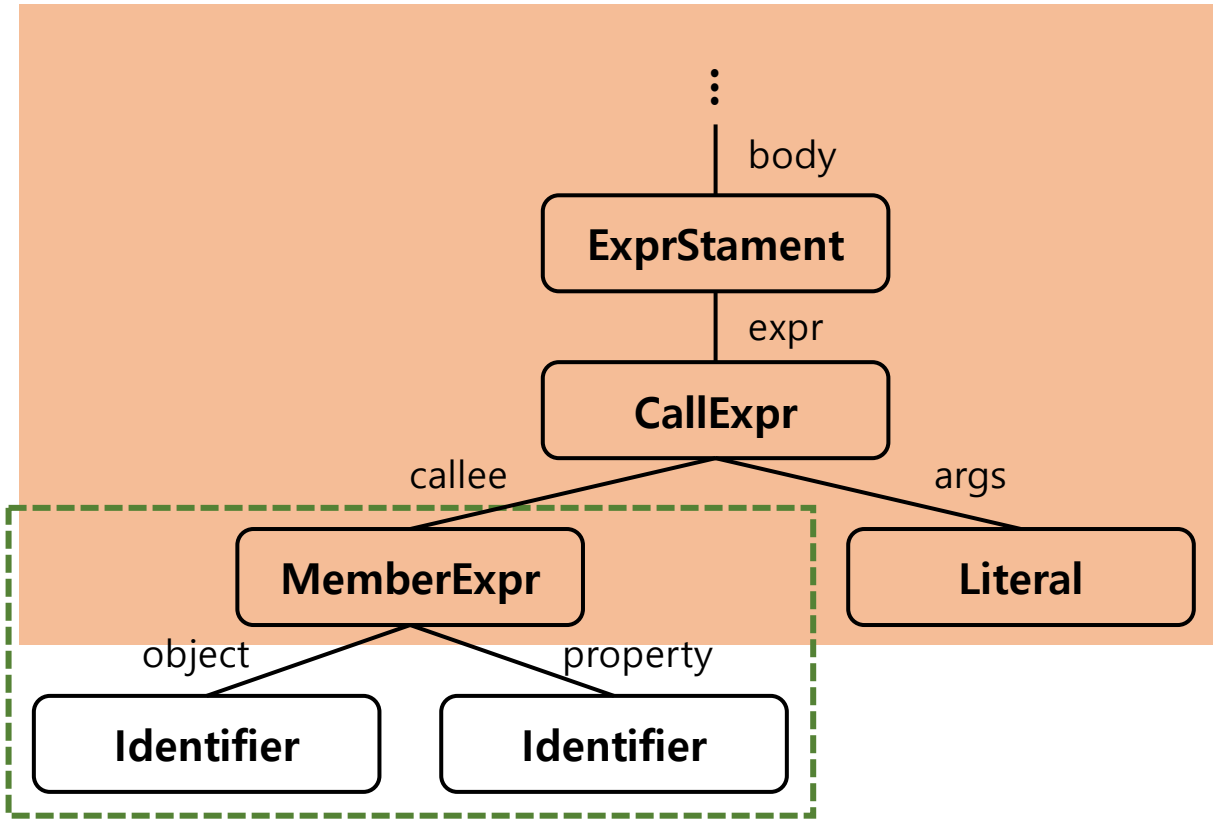
Trained LSTM model

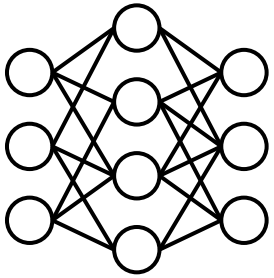
Randomly select one from the Top K fragments



0.76 0.12 0.001

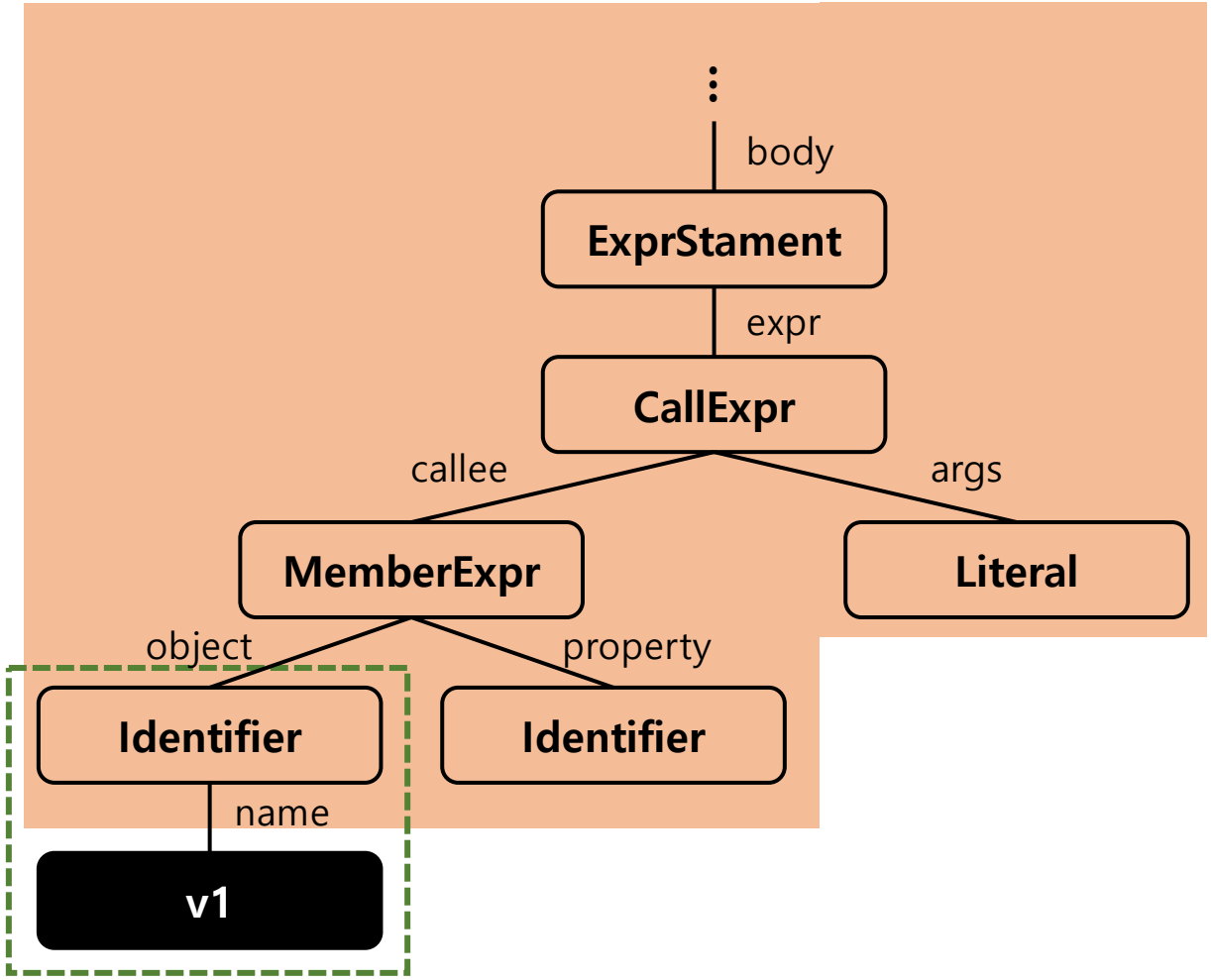
AST Mutation

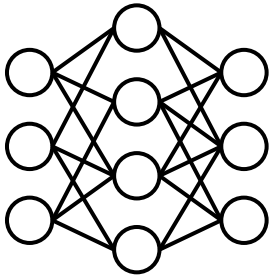




Trained LSTM model

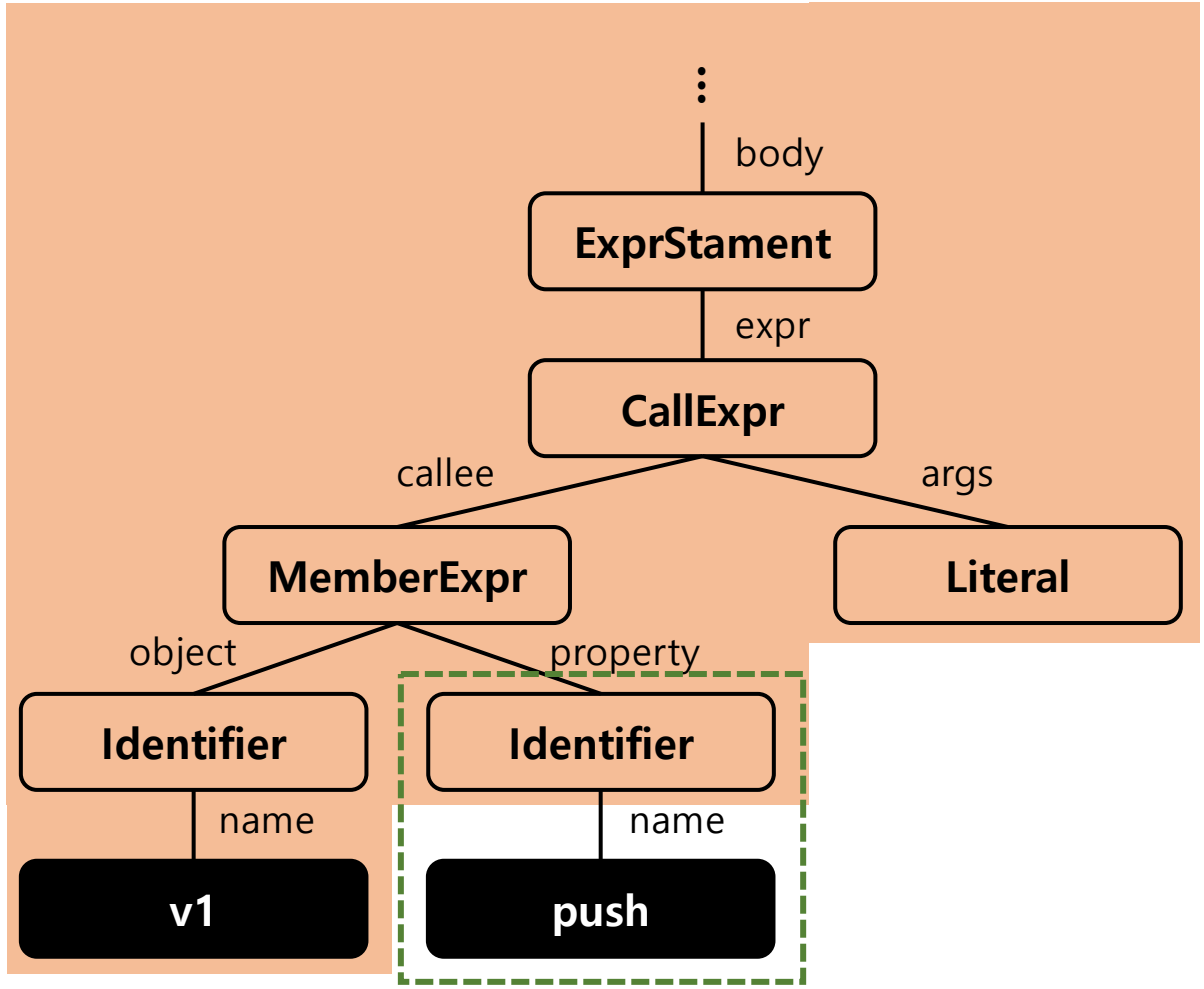
AST Mutation

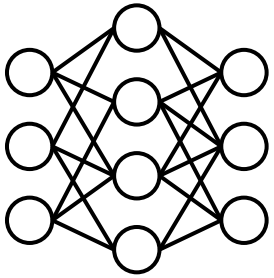




Trained LSTM model

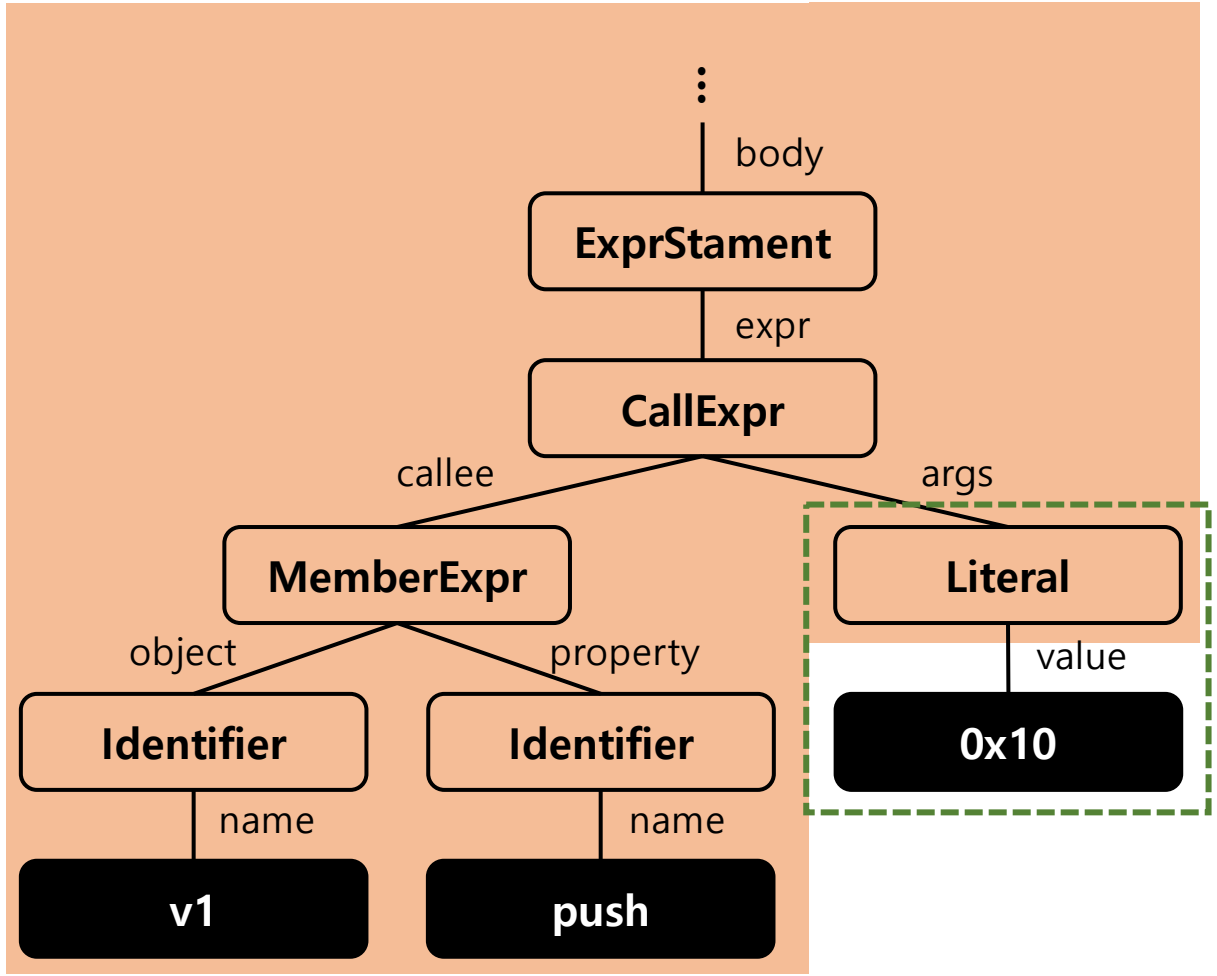
AST Mutation

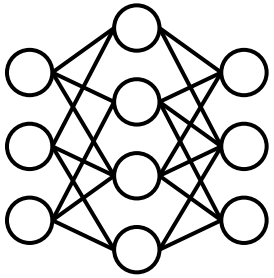




Trained LSTM model

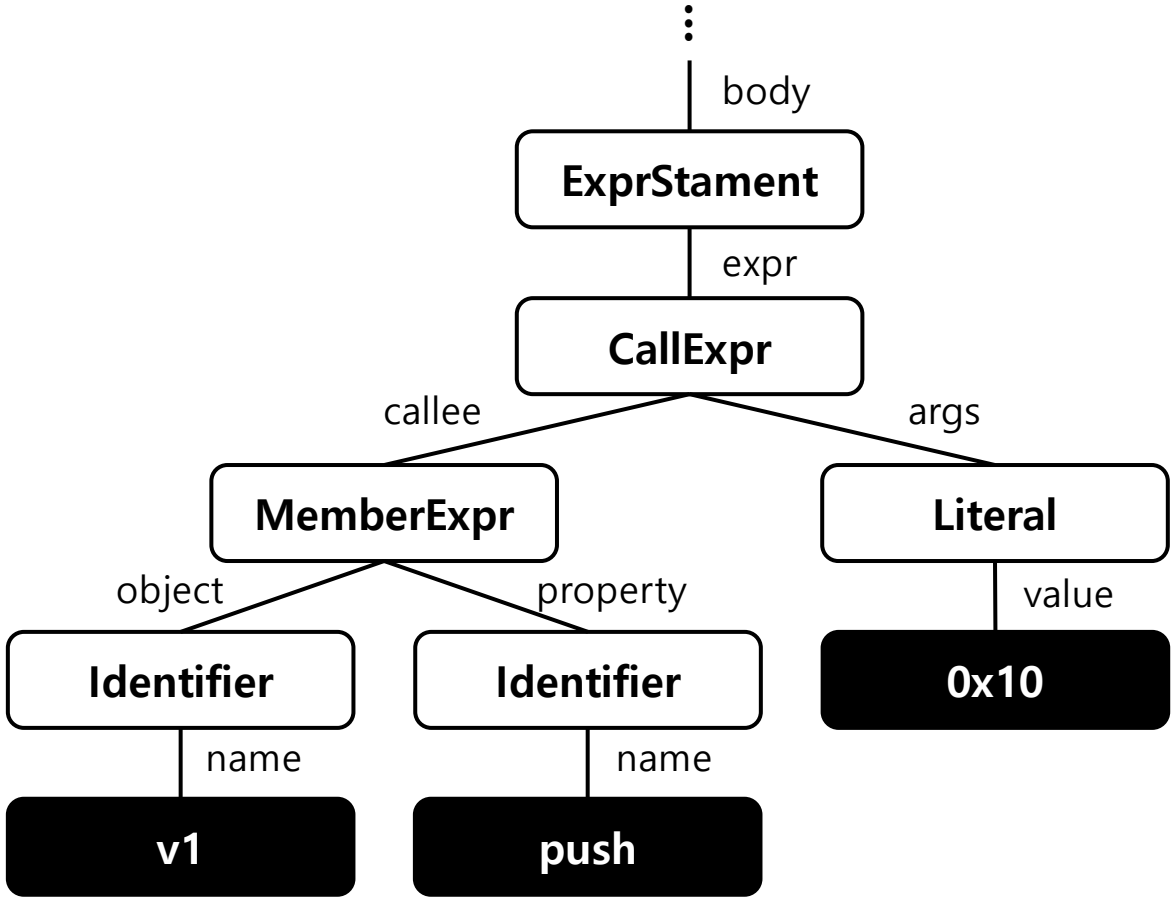
AST Mutation





Trained LSTM model

AST Mutation

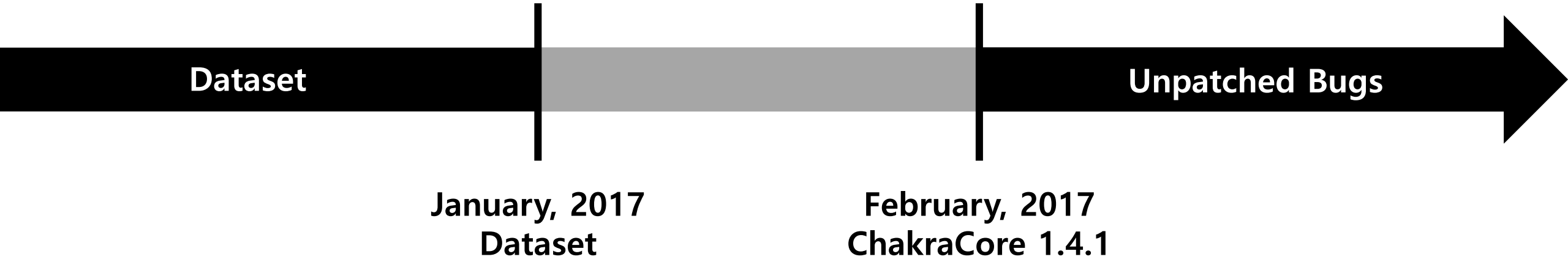


Mutated AST

```
var v0 = ‘Hello World’;
v1 = [];
f0();
function f0 () {
    v1.push(0x10);
    var v2 = 10;
}
```

Mutated JS

Experiment Setup



- Collected **33.5K** unique JS files
  - Regression tests from the repo of four major JS engines and Test262
  - PoCs of known CVEs
- Ran fuzzers against **ChakraCore 1.4.1**
- JS code triggering unpatched bugs is not in our training set!

KAIST

vs. State-of-the-art Fuzzers

- **72 hours x 5 trials**
  - CodeAlchemist: A **state-of-the-art** semantics-aware JS fuzzer, *NDSS'19*
  - jsfunfuzz: A JS fuzzer developed by Mozilla
  - IFuzzer: An evolutionary JS fuzzer, *ESORICS'16*

Metric	Build	# of Unique Crashes (Known CVEs)			
		Montage	**CodeAlchemist**	**jsfunfuzz**	**IFuzzer**
Median	Release	23 (7)	15 (4)	27 (3)	4 (1)
	Debug	49 (12)	26 (6)	27 (4)	6 (1)

The differences were **statistically significant** (p-value < 0.05)!

KAIST

A Sequence of Fragments vs. Tokens [1-3]

- **72 hours x 5 trials**
  - Token RNN: JS code mutation guided by a token-level LSTM model

Metric	Build	# of Unique Crashes (Known CVEs)	
		Montage	Token RNN
Median	Release	23 (7)	1 (0)
	Debug	49 (12)	3 (0)

[1] Cummins et al. Compiler fuzzing through deep learning (ISSTA'18).
[2] Godefroid et al. Learn&Fuzz: Machine learning for input fuzzing (ASE'17).
[3] Liu et al. DeepFuzz: Automatic generation of syntax valid C programs for fuzz testing (AAAI'19).

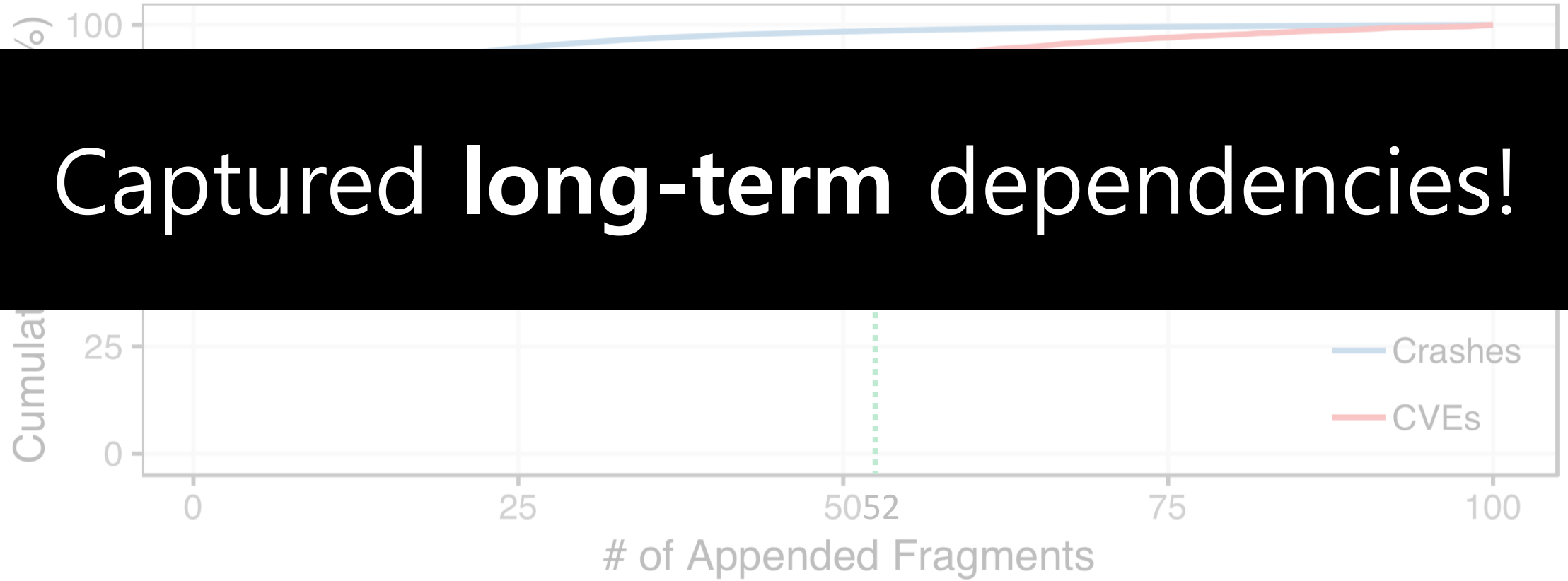
Is the LSTM Model Effective?

- **72 hours x 5 trials**
  - Random: Random fragment assembly without any model

Metric	Build	# of Unique Crashes (Known CVEs)	
		Montage	Random
Median	Release	23 (7)	12 (3)
	Debug	49 (12)	31 (7)

Is the LSTM Model Effective?

- The # of appended fragments to compose a new subtree



Captured **long-term** dependencies!

Finding Real-World Bugs

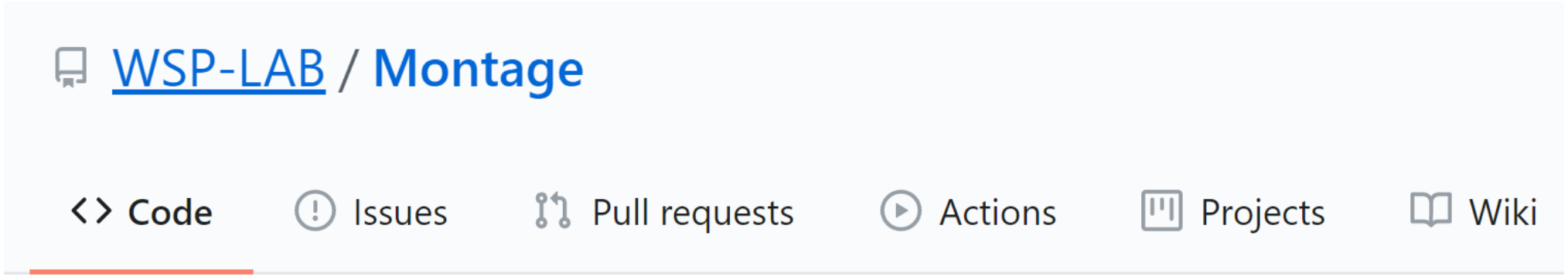
- **Four major JS engines for a total of 1.5 months**
  - Found **37 previously unknown bugs**
    - ➢ 34 bugs including **two CVEs** from ChakraCore 1.11.7
    - ➢ One bug from V8 7.4.0 (beta)
    - ➢ Two bugs including **one CVE** from JavaScriptCore 2.23.3
  - **26 of them were patched** at the time of writing

rewarded with **$5,000**!

Conclusion

- We proposed the **first neural network language model-guided JS engine fuzzer** and demonstrated its efficacy.
- We proposed a **novel approach of modeling JS code as a sequence of fragments** on which any prevalent language models can be trained without modification.
- Montage **outperformed state-of-the-art fuzzers** in the old version of ChakraCore.
- Montage found **37 previously unreported bugs** from the latest JS engines.

Open Science



https://github.com/WSP-LAB/Montage

For More Details

- Resolving reference errors
- Effect of parameters
- Effect of language models
- Case studies

Question?